3PC - Peer to Peer Pervasive Computing

Christian Becker, Marcus Handte, Gregor Schiele, Pedro Jose Marron
(Vorname.Nachnahme)@informatik.uni-stuttgart.de
Institut für Parallele und Verteilte Systeme
Universität Stuttgart

Abstract

Pervasive and ubiquitous computing environments gained increasing interest in the past years. Challenges in this particular research area are interoperability of services and support for application programmers to conquer the ever-changing execution environments. While most of the existing approaches rely on a centralized architecture responsible for a distinct spatial area (smart environments) we propose a middleware and component system that allows spontaneous interaction without relying on a central control.

Introduction

Pervasive Computing is characterized by the interaction of a multitude of highly heterogeneous devices, ranging from powerful general-purpose servers located in the infrastructure, to tiny mobile sensors, integrated in everyday objects. Devices are connected to each other on-the-fly using wireless communication technologies like Bluetooth, IEEE 802.11 or IrDA and share their functionality. A sensor could for instance use a nearby display to present its data to the user.

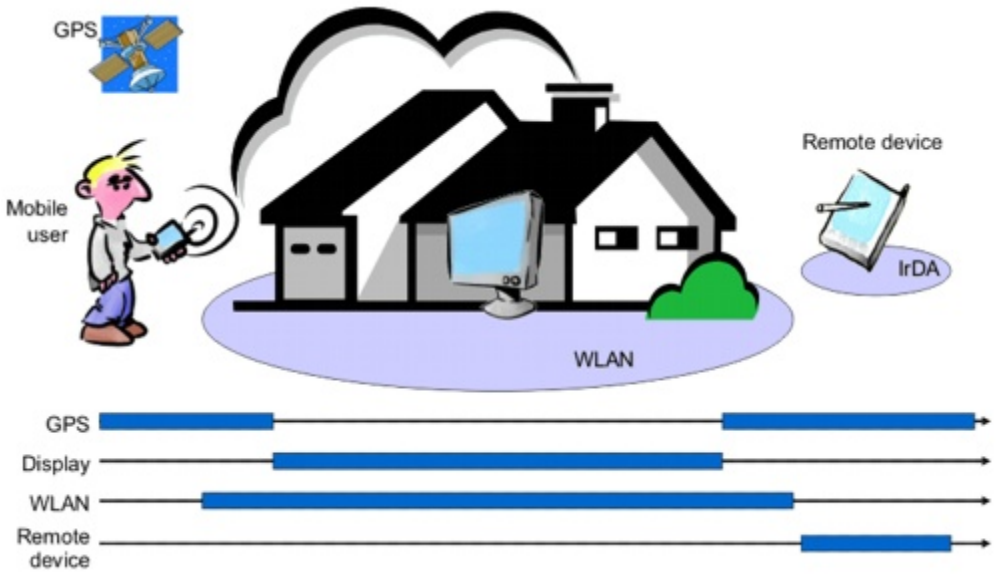

Figure 1: Scenario

Figure 1 sketches a situation where devices and functionality, e.g. the GPS sensor, are only available for distinct and prior undetermined periods of time. Devices cooperate in order to share their functionality – they are peers in a spontaneously networked environment.

Developing and executing applications in such environments is a non-trivial task. Apart from the device heterogeneity, the hardware and software resources, i.e. devices and services, available to an application are highly dynamic, due to factors like user mobility, fluctuating

network connectivity or changing physical context. This forces applications to adapt themselves constantly to their ever-changing execution environments. User-interaction, e.g. for adaptation control or administrative tasks, should be minimized, thus removing the user from the control loop.

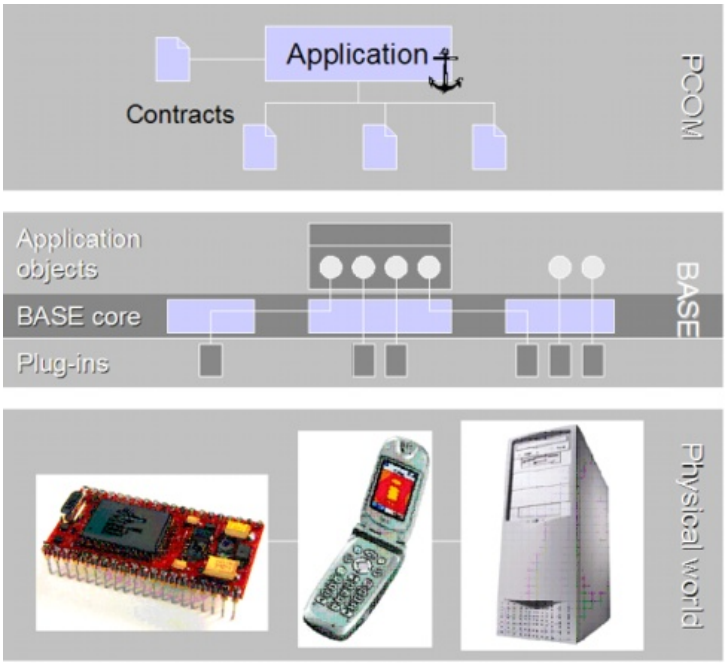

Figure 2: 3PC Architecture

In 3PC (cf. Figure 2), we investigate system software support for adaptive applications in peer to peer based spontaneous networks. We address both, heterogeneity of devices and communication technology. Our overall objectives are to provide system software that is minimal in its resource requirements in order to support resource-poor devices but flexible to allow the integration of all resources on powerful devices (BASE). Adaptation of applications is eased by an application model based on contractually specified components and a supporting component system (PCOM). Further research is conducted in the area of Power Aware Middleware (PAM) where discovery protocols and mediation of service interaction is scheduled in order to save energy.

In the following sections we will briefly describe BASE and PCOM and close this paper with an outlook on further research directions.

BASE – A Micro-Broker Based Middleware for Pervasive Computing

With BASE [1] we have developed a middleware based on a micro-broker design. The micro-broker mediates requests – so called invocations – between an application and local transport plug-ins or device wrappers. This design allows uniform access to fluctuating entities, such as remote services or device capabilities, like a GPS receiver.

Figure 3 sketches the core architecture of BASE. The programming model of BASE is object-oriented. Clients and services make use of proxy objects (stubs and skeletons) to interact with each other. The micro-broker relies on a federated registry. Via discovery mechanisms provided for each transport protocol it detects devices that are in communication range. Applications that require a service can query the local proxy of the federated registry. Besides the registry, BASE offers a signaling mechanism that detects the unavailability of services.

BASE has been implemented in Java using the Java 2 Microedition and supports different profiles, such as the CDC and the CLDC. Together with a small memory footprint ranging from 90 to 120 KB, this allows the deployment on small embedded systems such as the JStamp (http//:www.jstamp.org). However, the programming model offered by BASE requires programmers to deal manually with adaptation for each single resource that may change its availability or quality. For instance, when a used service is reselected, BASE does not offer any support for transparent transfer of state. Thus, only stateless services can be reselected without programming extra code that provides this functionality at the application level. Clearly, this is not desirable. By designing PCOM we aimed at an abstraction that allows high-level adaptation of applications by enhancing the system support provided by BASE, its underlying middleware.

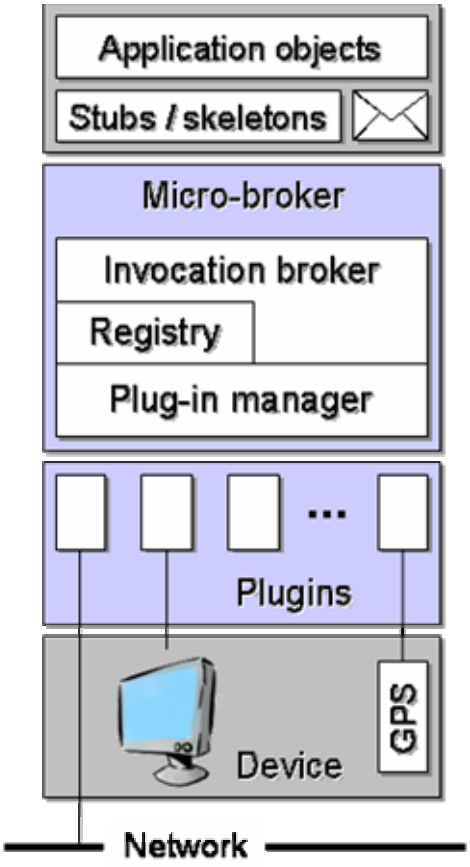

Figure 3: BASE Architecture

PCOM – A Component System for Pervasive Computing

Automatic adaptation of an application requires the system to have knowledge about the application's architecture, its possible configurations, and the user's preferences regarding these configurations. In order to reflect the dependencies of services with respect to the underlying platform, e.g. memory requirements or some device such as a GPS receiver, and other services we have designed a component model based on BASE named PCOM [2].

Components in PCOM specify an offer via a bidirectional interface. This allows components to receive and process requests as well as to emit signals to their communication partner allowing for asynchronous operation (cf. Figure 4). The requirements of a component and its offers are manifested in a contract that captures not only the functional, but also non-functional properties (cf. Figure 4).

An application is composed of components that only rely on their contractually specified dependencies. One distinguished component – the so-called application anchor – manifests and identifies the application itself. An application can be executed when all of its components can be instantiated, i.e. all dependencies are fulfilled.

In PCOM we support adaptation with three different mechanisms. First, components that fulfill the requirements of another component can be automatically determined. If more than one component can fulfill the requirements the system chooses one. At the moment, users specify their preferences using a simple metric that is part of the contracts. However, the integration of metrics and the design of selection strategies for alternative application configurations is one of the subjects of our current research. The selection mechanism is used whenever an application is initialized or if a used component becomes unavailable and an alternative has to be determined. Second, a component is enabled to stop its operation in cases where the quality or availability of one of its used components changes. This escalates adaptation to the component that has been using the stopped component. Third, a component may modify its contract according to changes in its execution environment. As such modifications require knowledge about the implementation of the component, they must be provided by the component programmer. Changes regarding contracts may also lead to a reselection or escalation.

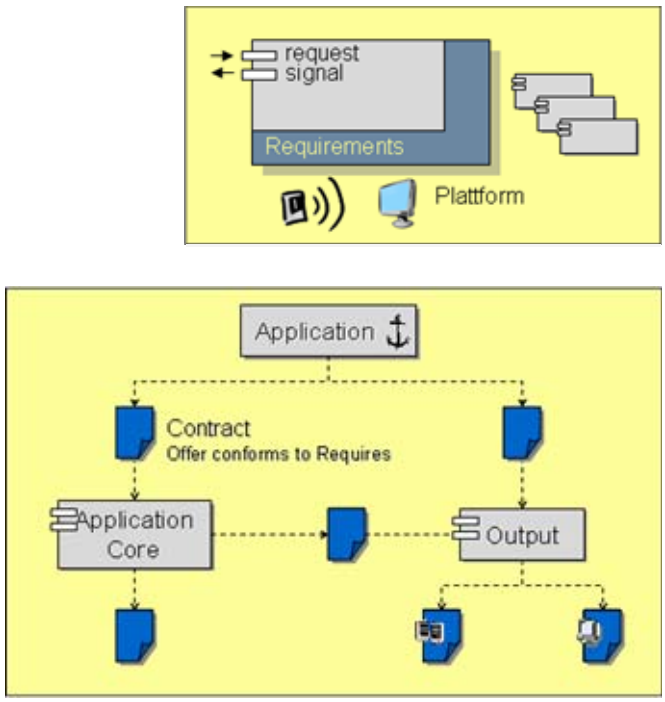

Figure 4: Components (top) and Application Model (bottom)

PCOM components are executed in a component container that is implemented as a BASE service. As a result, components make use of the flexible transport selection and discovery mechanisms of BASE. The container is responsible for keeping track of available components in the execution environment. Therefore, a container propagates all contracts offered by its local components. If an application is started, the container tries to resolve all dependencies by matching contracts from available devices.

The PCOM component model was designed to meet the same requirements as BASE, i.e. allow an installation even on resource-poor devices. The overhead of PCOM is about 30-40 KB and PCOM also requires J2ME in its minimum configuration.

Further Research Directions

So far, we have realized an exemplary architecture for system support in peer to peer based Pervasive Computing environments. The application model introduced by PCOM provides mechanisms that can automate the selection of alternatives. However, the design of selection strategies in cases where multiple sets of components can be used to satisfy the dependencies of a component is still an open research challenge. Complex but interesting conflicts arise especially in cases when multiple applications from different users are executed in a single environment.

Another open research challenge in the field of peer to peer based environments that we are investigating results from the power limitations of mobile battery-operated devices. In 3PC we are currently developing a set of service discovery and interaction protocols that take these restrictions into account and help to reduce the energy consumption of mobile devices at the middleware level.

Literature

[1] C. Becker, G. Schiele, H. Gubbels, K. Rothermel "BASE - A Micro-broker-based Middleware For Pervasive Computing" in Proceedings of the IEEE International Conference on Pervasive Computing and Communication (PerCom), Fort Worth, USA

[2] C. Becker, M. Handte, G. Schiele, K. Rothermel “PCOM - A Component System for Pervasive Computing“, to appear in 2nd IEEE International Conference on Pervasive Computing and Communication (PerCom 04), Orlando, USA

[3] M. Handte, C. Becker, G. Schiele “Experiences: Extensibility and Flexibility in BASE” Workshop System Support for Ubiquitous Computing (UbiSys) at UbiComp, Seattle/USA, 2003

[4] C. Becker, G. Schiele “BASE: A Minimal yet Extensible Platform for Pervasive Computing“, In Proceedings of the International Conference on „Tales of the Disappearing Computer“, Santorin/ Greece, 2003

[5] C. Becker, G. Schiele “Application Adaption Requirements and their Support in Pervasive Computing“ In Proceedings of the 3rd International Workshop on Distributed Auto-adaptive and Reconfigurable Systems, ICDCS 2003, Providence/USA, 2003